1. (Original) A background event buffer manager (BEBM) for ordering and accounting for events in a data processing system having a processor, the BEBM comprising:

a port for receiving event identifications (IDs) from a device;

a queuing function enabled for queuing event IDs received; and

a notification function for notifying the processor of queued event IDs;

characterized in that the BEBM handles all event ordering and accounting for the processor.

2. (Original) The BEBM of claim 1 wherein the queuing function queues event IDs by type, and by event priority within type queues, and also associates an acknowledgment (ack) with each event.

3. (Original) The BEBM of claim 2 wherein the ack may have multiple states, and acks for events first queued are set in a "processor unaware" state.

4. (Original) The BEBM of claim 3 wherein, after notification to the processor the ack state for the queued event is changed to a state of "processor aware".

5. (Original) The BEBM of claim 4 comprising a function for receiving notification from the processor of processing completed on an event, and wherein, upon receiving such a notification the state of the ack for the event is changed to "ready".

6. (Original) The BEBM of claim 5 further comprising a function for sending acks in ready state back to the device that originally sent the event associated with the ack, and buffering the process of sending the acks.

7. (Original) The BEBM of claim 2 wherein the processor is notified of events in order of type priority first, and then by event priority within type.

8. (Original) The BEBM of claim 1 wherein the events are arrival of packets to be processed in a network packet router.

9. (Original) A data processing system, comprising:

a processor;

a memory coupled to the processor; and

a background event buffer manager BEBM coupled to the processor, the BEBM including a port for receiving event identifications (IDs) from a device, a queuing function enabled for queuing event IDs received, and a notification function for notifying the processor of queued event IDs;

characterized in that the BEBM handles all event ordering and accounting for the processor.

10. (Original) The data processing system of claim 9 wherein the queuing function queues event IDs by type, and by event priority within type queues, and also associates an acknowledgment (ack) with each event.

11. (Original) The data processing system of claim 10 wherein the ack may have multiple states, and acks for events first queued are set in a "processor unaware" state.

12. (Original) The data processing system of claim 11 wherein, after notification to the processor the ack state for the queued event is changed to a state of "processor aware".

13. (Original) The data processing system of claim 12 comprising a function for receiving notification from the processor of processing completed on an event, and wherein, upon receiving such a notification the state of the ack for the event is changed to "ready".

14. (Original) The data processing system of claim 13 further comprising a function for sending acks in ready state back to the device that originally sent the event associated with the ack, and for buffering the sending process.

15. (Original) The data processing system of claim 10 wherein the processor is notified of events in order of type priority first, and then by event priority within type.

16. (Original) The data processing system of claim 9 wherein the events are arrival of packets to be processed in a network packet router, and the system is a packet processing engine.

17. (Original) A network packet router, comprising:

an input/output (I/O) device for receiving and sending packets on the network;

a processor;

a memory coupled to the processor; and

a background event buffer manager BEBM coupled to the processor, the BEBM including a port for receiving event identifications (IDs) of arriving packets from the I/O device, a queuing function enabled for queuing packet IDs received, and a notification function for notifying the processor of queued packet IDs for processing;

characterized in that the BEBM handles all event ordering and accounting for the processor.

18. (Original) The network packet router of claim 17 wherein the queuing function queues event IDs by type, and by event priority within type queues, and also associates an acknowledgment (ack) with each event.

19. (Original) The network packet router of claim 18 wherein the ack may have multiple states, and acks for events first queued are set in a "processor unaware" state.

20. (Original) The network packet router of claim 19 wherein, after notification to the processor the ack state for the queued event is changed to a state of "processor aware".

21. (Original) The network packet router of claim 20 comprising a function for receiving notification from the processor of processing completed on an event, and wherein, upon receiving such a notification the state of the ack for the event is changed to "ready".

22. (Original) The network packet router of claim 21 further comprising a function for sending acks in ready state back to the device that originally sent the event associated with the ack, and for buffering the sending process.

[22] 23. (Currently Amended) The network packet router of claim 18 wherein the processor is notified of events in order of type priority first, and then by event priority within type.

[23] 24. (Currently Amended) The network packet router of claim 17 wherein the events are arrival of packets to be processed in a network packet router, and the system is a packet processing engine.

[24] 25. (Currently Amended) A method for ordering and accounting for events in a data processing system having a processor, the method comprising steps of:

(a) generating event identifications (IDs) by a device;

(b) sending the event IDs to a background event buffering manager (BEBM) by the device;

(c) queuing event IDs received by the BEBM; and

(d) notifying the processor by the BEBM of events queued for processing, such that the BEBM handles all event ordering and accounting for the processor.

[25] 26. (Currently Amended) The method of claim 24 wherein, in step (c), event IDs are queued by type, and by event priority within type queues, and an acknowledgment (ack) is associated with each event.

[26] 27. (Currently Amended) The method of claim 25 wherein the ack may have multiple states, and acks for events first queued are set in a "processor unaware" state.

[27] 28. (Currently Amended) The method of claim 26 wherein, after notification to the processor the ack state for the queued event is changed to a state of "processor aware".

[28] 29. (Currently Amended) The method of claim 27 comprising a further step for receiving, at the BEBM, notification from the processor of processing completed on an event, and wherein, upon receiving such a notification the state of the ack for the event is changed to "ready".

[29] 30. (Currently Amended) The method of claim 28 further comprising a step for sending acks in ready state back to the device that originally sent the event associated with the ack, and for buffering the sending process.

[30] 31. (Currently Amended) The method of claim 25 wherein the processor is notified of events in order of type priority first, and then by event priority within type.

[31] 32. (Currently Amended) The method of claim 24 wherein the events are arrival of packets to be processed in a network packet router, and the system is a packet processing engine.